

[illegible]

★★

★★


```
1 0001 0 MODULE FIND (
2 0002 0
3 0003 0 LANGUAGE (BLISS32),
4 0004 0 IDENT = 'V04-000'
5 0005 1 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This routine performs a find operation on the indicated directory.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1 STARLET operating system, including privileged system services
42 0042 1 and internal exec routines.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 3-Jan-1978 13:37
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 V03-007 ACG0408 Andrew C. Goldstein, 21-Mar-1984 12:04
52 0052 1 Make APPLY_RVN and DEFAULT_RVN macros
53 0053 1
54 0054 1 V03-006 ACG0398 Andrew C. Goldstein, 14-Feb-1984 20:45
55 0055 1 Fix tie-off of NOP find, done wrong in ACG0354
56 0056 1
57 0057 1 V03-005 CDS0002 Christian D. Saether 18-Jan-1984
```

FIND
V04-000

M 10
16-Sep-1984 00:31:12
14-Sep-1984 12:30:27

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]FIND.B32;1

Page 2
(1)

```

58      0058 1  Modify interface to APPLY_RVN and DEFAULT_RVN.
59      0059 1
60      0060 1  V03-004 CDS0001      Christian D. Saether      29-Dec-1983
61      0061 1      Use L_NORM linkage and BIND_COMMON macro.
62      0062 1
63      0063 1  V03-003 LMP0166      L. Mark Pilant,          28-Oct-1983  19:13
64      0064 1      Correct a bug that caused execute access to grant write access
65      0065 1      to directories during a create-if operation.
66      0066 1
67      0067 1  V03-002 ACG0354      Andrew C. Goldstein,      15-Sep-1983  10:52
68      0068 1      Tie off FIND with FINDFID and no result string
69      0069 1
70      0070 1  V03-001 ACG0323      Andrew C. Goldstein,      25-Mar-1983  18:42
71      0071 1      Add optional result name arguments
72      0072 1
73      0073 1  V02-006 ACG0259      Andrew C. Goldstein,      26-Jan-1982  19:13
74      0074 1      Add mode arg to REMOVE
75      0075 1
76      0076 1  V02-005 ACG0228      Andrew C. Goldstein,      25-Nov-1981  19:16
77      0077 1      Support execute protection on directories
78      0078 1
79      0079 1  V02-004 ACG0223      Andrew C. Goldstein,      17-Nov-1981  21:52
80      0080 1      Support modification of version limit
81      0081 1
82      0082 1  V02-003 ACG0208      Andrew C. Goldstein,      12-Nov-1981  11:32
83      0083 1      Add segmented directory record support
84      0084 1
85      0085 1  V02-002 ACG0167      Andrew C. Goldstein,      16-Apr-1980  19:26
86      0086 1      Previous revision history moved to F11B.REV
87      0087 1  **
88      0088 1
89      0089 1
90      0090 1  LIBRARY 'SYS$LIBRARY:LIB.L32';
91      0091 1  REQUIRE 'SRC$:FCPDEF.B32';
```

GE
VO


```

93      1082 1 GLOBAL ROUTINE FIND (ABD, FIB, FIND_MODE, RESLEN_ARG, RESULT_ARG) : L_NORM NOVALUE =
94      1083 1
95      1084 1 |++
96      1085 1
97      1086 1 FUNCTIONAL DESCRIPTION:
98      1087 1
99      1088 1     This routine performs a FIND operation on the indicated directory.
100     1089 1
101     1090 1 CALLING SEQUENCE:
102     1091 1     FIND (ARG1, ARG2, ARG3, ARG4, ARG5)
103     1092 1
104     1093 1 INPUT PARAMETERS:
105     1094 1     ARG1: address of descriptor list in buffer packet
106     1095 1     ARG2: address of FIB
107     1096 1     ARG3: 0 to just do a find
108     1097 1           1 to remove the found entry
109     1098 1           2 to set version limit in entry
110     1099 1           4 to locate file and access directory for write
111     1100 1
112     1101 1 IMPLICIT INPUTS:
113     1102 1     NONE
114     1103 1
115     1104 1 OUTPUT PARAMETERS:
116     1105 1     ARG2: file ID and context returned in FIB
117     1106 1     ARG4: (optional) length of found file name
118     1107 1     ARG5: (optional) found filename
119     1108 1
120     1109 1 IMPLICIT OUTPUTS:
121     1110 1     NONE
122     1111 1
123     1112 1 ROUTINE VALUE:
124     1113 1     NONE
125     1114 1
126     1115 1 SIDE EFFECTS:
127     1116 1     Directory LRU may be altered
128     1117 1     directory blocks read
129     1118 1     resultant string written into buffer packet
130     1119 1
131     1120 1 --
132     1121 1
133     1122 2 BEGIN
134     1123 2
135     1124 2 MAP
136     1125 2     ABD          : REF BBLOCKVECTOR [,ABD$C_LENGTH],
137     1126 2             ! descriptor list arg
138     1127 2     FIB      : REF BBLOCK;      ! FIB argument
139     1128 2
140     1129 2 LOCAL
141     1130 2     COUNT,          ! count of name string
142     1131 2     NAME_DESC    : BBLOCK [FND_LENGTH], ! file name descriptor
143     1132 2     PREV_DESC   : BBLOCK [FND_LENGTH], ! previous name descriptor
144     1133 2     NAME_BUFFER  : VECTOR [FILENAME_LENGTH, BYTE],
145     1134 2             ! Buffer to hold parsed file name string
146     1135 2     PREV_BUFFER  : VECTOR [FILENAME_LENGTH, BYTE],
147     1136 2             ! Buffer to hold predecessor name string
148     1137 2     RESULT_STRING : VECTOR [FILENAME_LENGTH+6, BYTE],
149     1138 2             ! Buffer to build result name string

```



```
150      1139 2      RESULT_LENGTH,      | length of above
151      1140 2      START_BLOCK,      | block number to start search
152      1141 2      START_REC,      | record number to start search
153      1142 2      START_VER,      | version entry to start search
154      1143 2      P      : REF BBLOCK, | pointer to directory record
155      1144 2      VBN;      | VBN of directory block processed
156      1145 2
157      1146 2 BIND_COMMON;
158      1147 2
159      1148 2 DIR_CONTEXT_DEF;
160      1149 2
161      1150 2 EXTERNAL ROUTINE
162      1151 2      PMS_START_SUB      : L_NORM ADDRESSING_MODE (GENERAL),
163      1152 2      | start subfunction metering
164      1153 2      PMS_END_SUB      : L_NORM ADDRESSING_MODE (GENERAL),
165      1154 2      | end subfunction metering
166      1155 2      COPY_NAME      : L_NORM,      | copy file name to result string
167      1156 2      DIR_ACCESS      : L_NORM,      | access the directory
168      1157 2      PARSE_NAME      : L_NORM,      | parse file name string
169      1158 2      DIR_SCAN      : L_NORM,      | search the directory
170      1159 2      NEXT_REC      : L_NORM,      | get next directory record
171      1160 2      RETURN_DIR      : L_NORM,      | return file name to user
172      1161 2      MARK_DIRTY      : L_NORM,      | mark buffer for write-back
173      1162 2      NEXT_DIR_REC      : L_NORM,      | read next directory record
174      1163 2      REMOVE      : L_NORM;      | remove directory entry
175      1164 2
176      1165 2
177      1166 2 ! Start metering for this subfunction.
178      1167 2 !
179      1168 2
180      1169 2 PMS_START_SUB (PMS_FIND);
181      1170 2
182      1171 2 ! If this is an operation on a spooled device, noop it and return a file ID
183      1172 2 ! of -2, -2 with success.
184      1173 2 !
185      1174 2
186      1175 2 IF .CLEANUP_FLAGS[CLF_SPOOLFILE]
187      1176 2 THEN
188      1177 3 BEGIN
189      1178 3 FIB[FIB$W_FID_NUM] = -2;
190      1179 3 FIB[FIB$W_FID_SEQ] = -2;
191      1180 3 FIB[FIB$W_FID_RVN] = 0;
192      1181 3 FIB[FIB$B_FID_NMX] = -1;
193      1182 3 KERNEL_CALL (COPY_NAME, .ABD);
194      1183 3 RETURN;
195      1184 2 END;
196      1185 2
197      1186 2 ! Parse the file name.
198      1187 2 !
199      1188 2
200      1189 2 PARSE_NAME (NAME_DESC, NAME_BUFFER, .ABD[ABD$C_NAME, ABD$W_COUNT],
201      1190 2      ABD[ABD$C_NAME, ABD$W_TEXT] + .ABD[ABD$C_NAME, ABD$W_TEXT] + 1, .FIB[FIB$W_NMCTL]);
202      1191 2
203      1192 2 ! Access the directory. We need write access if this is a remove or set
204      1193 2 ! version operation, read access if there are any wild cards, and
205      1194 2 ! execute access if just a simple lookup.
206      1195 2 !
```



```
207 1196 2
208 1197 2 DIR_ACCESS (.FIB,
209 1198 2 (IF .FIND MODE NEQ 0
210 1199 2 THEN WRITE_ACCESS
211 1200 2 ELSE IF .NAME_DESC[FND_WILD]
212 1201 2 THEN READ_ACCESS
213 1202 2 ELSE EXEC_ACCESS));
214 1203 2
215 1204 2 ! If the lookup will have no further effect (FINDFID, no remove, and
216 1205 2 ! no result string) then all we wanted was a protection check on the
217 1206 2 ! directory, so stop now.
218 1207 2
219 1208 2
220 1209 2 IF .FIND MODE EQL 0
221 1210 2 AND .FIB[FIB$V_FINDFID]
222 1211 2 AND .ABD[ABD$C_RES, ABD$W_COUNT] EQL 0
223 1212 2 THEN RETURN;
224 1213 2
225 1214 2 START_BLOCK = 0; ! assume search from start
226 1215 2 START_REC = 0;
227 1216 2 START_VER = 0;
228 1217 2 DIR_RECORD = 0;
229 1218 2 DIR_PRED = 0;
230 1219 2 LAST_ENTRY[0] = 0;
231 1220 2
232 1221 2 ! Default the RVN of the supplied file ID to 0 if current volume, to
233 1222 2 ! allow the search for file ID to work.
234 1223 2
235 1224 2
236 1225 2 DEFAULT_RVN (FIB[FIB$W_FID_RVN], .CURRENT_RVN);
237 1226 2
238 1227 2 ! If this is a wild card operation (i.e., if the wild card context is
239 1228 2 ! nonzero), position to the indicated record. This is done with the
240 1229 2 ! positional context and the supplied resultant name string, if any.
241 1230 2
242 1231 2
243 1232 2 IF .FIB[FIB$L_WCC] NEQ 0
244 1233 2 THEN
245 1234 2 BEGIN
246 1235 2 START_BLOCK = .(FIB[FIB$L_WCC])<6,10>;
247 1236 2 START_REC = .(FIB[FIB$L_WCC])<0,6> - 1;
248 1237 2 COUNT = MINU (.ABD[ABD$C_RES, ABD$W_COUNT],
249 1238 2 IF .ABD[ABD$C_RES, ABD$W_COUNT] GEQU 2
250 1239 2 THEN .(ABD[ABD$C_RES, ABD$W_TEXT] +
251 1240 2 .ABD[ABD$C_RES, ABD$W_TEXT] + 1)<0,16>
252 1241 2 ELSE 0
253 1242 2 );
254 1243 2 PARSE_NAME (PREV_DESC, PREV_BUFFER, .COUNT, ABD[ABD$C_RES, ABD$W_TEXT]
255 1244 2 + .ABD[ABD$C_RES, ABD$W_TEXT] + 1, 0);
256 1245 2 IF .PREV_DESC[FND_WILD]
257 1246 2 THEN ERR_EXIT (SS$_BADFILENAME);
258 1247 2
259 1248 2 ! If no resultant string is supplied (noted by a 0 count), position by
260 1249 2 ! scanning over the indicated number of records in the indicated block.
261 1250 2 ! Then, if there are no wild cards in the version, scan to the last version.
262 1251 2
263 1252 2
```



```
264 1253 3 IF .COUNT EQL 0
265 1254 3 THEN
266 1255 4 BEGIN
267 1256 4 PREV_DESC[FND_FID] = 1;
268 1257 4 DIR_SCAN (PREV_DESC, UPLIT WORD (-1, -1, -1), .START_BLOCK, 0, 0, 0, .START_REC);
269 1258 4 IF NOT .NAME_DESC[FND_WILD_VER]
270 1259 4 THEN
271 1260 5 BEGIN
272 1261 5 PREV_DESC[FND_FLAGS] = FIBSM_WILD;
273 1262 5 PREV_DESC[FND_COUNT] = 3;
274 1263 5 PREV_DESC[FND_STRING] = UPLIT BYTE ('*.*');
275 1264 5 PREV_DESC[FND_VERSION] = -32768;
276 1265 5 DIR_SCAN (PREV_DESC, 0, .DIR_VBN-1, .DIR_ENTRY, .DIR_VERSION, .DIR_PRED, -1);
277 1266 4 END;
278 1267 4 START_VER = .DIR_VERSION + DIRSC_VERSION;
279 1268 4 DIR_RECORD = .DIR_RECORD + 1;
280 1269 4 END
281 1270 4
282 1271 4 ! If a resultant string is supplied, search the indicated block for the
283 1272 4 ! given entry. If the search fails immediately (no records are
284 1273 4 ! traversed), search again from the start. If the version is not wild,
285 1274 4 ! we search for the oldest version so we are positioned at the start of the
286 1275 4 ! next name. Thus we are left positioned at the record, or where it
287 1276 4 ! used to be.
288 1277 4 !
289 1278 4
290 1279 3 ELSE
291 1280 4 BEGIN
292 1281 4 IF NOT .NAME_DESC[FND_WILD_VER]
293 1282 4 THEN PREV_DESC[FND_VERSION] = -32768;
294 1283 4 IF DIR_SCAN (PREV_DESC, 0, .START_BLOCK, 0, 0, 0, -1)
295 1284 4 THEN
296 1285 5 BEGIN
297 1286 5 START_VER = .DIR_VERSION + DIRSC_VERSION;
298 1287 5 DIR_RECORD = .DIR_RECORD + 1;
299 1288 5 END
300 1289 4 ELSE IF
301 1290 5 (
302 1291 5 IF .DIR_VBN - 1 LEQU .START_BLOCK
303 1292 5 AND .DIR_RECORD EQL 0
304 1293 5 THEN DIR_SCAN (PREV_DESC, 0, 0, 0, 0, 0, -1)
305 1294 5 ELSE 0
306 1295 5 )
307 1296 4 THEN
308 1297 5 BEGIN
309 1298 5 START_VER = .DIR_VERSION + DIRSC_VERSION;
310 1299 5 DIR_RECORD = .DIR_RECORD + 1;
311 1300 5 END
312 1301 4 ELSE START_VER = .DIR_VERSION;
313 1302 3 END;
314 1303 3
315 1304 3 START_REC = .DIR_ENTRY;
316 1305 3 START_BLOCK = .DIR_VBN - 1;
317 1306 2 END;
318 1307 2
319 1308 2 ! Now actually search for the desired directory entry.
320 1309 2 !
```

: R
:
:


```
321 1310 2
322 1311 2 IF NOT DIR_SCAN (NAME_DESC, FIB[FIB$W_FID], .START_BLOCK, .START_REC, .START_VER, .DIR_PRED, -1)
323 1312 2 THEN
324 1313 2     IF .FIB[FIB$L_WCC] EQL 0
325 1314 2     THEN ERR_EXIT (SS$NOSUCHFILE)
326 1315 2     ELSE
327 1316 2         BEGIN
328 1317 2         FIB[FIB$L_WCC] = 0;
329 1318 2         ERR_EXIT (SS$NOMOREFILES);
330 1319 2         END;
331 1320 2
332 1321 2 ! Extract the file ID and context of the found entry and return the resultant
333 1322 2 ! string.
334 1323 2 !
335 1324 2
336 1325 2 CH$MOVE (FID$C_LENGTH, DIR_VERSION[DIR$W_FID], FIB[FIB$W_FID]);
337 1326 2 APPLY RVN (FIB[FIB$W_FID_RVN], .CURRENT_RVN);
338 1327 2 IF .NAME_DESC[FND_WI[D]
339 1328 2 OR .NAME_DESC[FND_WILD_VER]
340 1329 2 OR .FIB[FIB$V_WILD]
341 1330 2 THEN
342 1331 2     FIB[FIB$L_WCC] = (.DIR_VBN-1) ^ 6 + .DIR_RECORD + 1
343 1332 2 ELSE
344 1333 2     FIB[FIB$L_WCC] = 0;
345 1334 2
346 1335 2 KERNEL CALL (RETURN_DIR, RESULT_LENGTH, RESULT_STRING, .ABD);
347 1336 2 IF ACTUALCOUNT GEQU 5
348 1337 2 THEN
349 1338 2     BEGIN
350 1339 2     .RESLEN_ARG = .RESULT_LENGTH;
351 1340 2     CH$MOVE (.RESULT_LENGTH, RESULT_STRING, .RESULT_ARG);
352 1341 2     END;
353 1342 2
354 1343 2 ! If we are asked to set the version limit, do so. We iterate through
355 1344 2 ! multiple directory records if necessary. Then return the version
356 1345 2 ! limit, regardless.
357 1346 2 !
358 1347 2
359 1348 2 IF .FIND MODE EQL 2
360 1349 2 AND .VERSION_COUNT EQL 0
361 1350 2 THEN
362 1351 2     BEGIN
363 1352 2     P = .DIR_ENTRY;
364 1353 2     VBN = .DIR_VBN;
365 1354 2     DO
366 1355 2         BEGIN
367 1356 2         P[DIR$W_VERLIMIT] = .FIB[FIB$W_VERLIMIT];
368 1357 2         MARK DIRTY (.P);
369 1358 2         P = NEXT_DIR_REC (.P, VBN);
370 1359 2         END
371 1360 2     UNTIL .P EQL 0;
372 1361 2     END;
373 1362 2 FIB[FIB$W_VERLIMIT] = .VERSION_LIMIT;
374 1363 2
375 1364 2 ! If this is a REMOVE function, do so.
376 1365 2 !
377 1366 2
```

FIND
V04-000

F 11
16-Sep-1984 00:31:12
14-Sep-1984 12:30:27

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]FIND.B32;1

Page 8
(2)

**F

```

: 378      1367 2 IF .FIND MODE
: 379      1368 2 THEN REMOVE (0);
: 380      1369 2
: 381      1370 2 ! Stop metering of this subfunction.
: 382      1371 2 !
: 383      1372 2
: 384      1373 2 PMS_END_SUB ();
: 385      1374 2
: 386      1375 1 END;
```

! end of routine FIND

				.TITLE	FIND	
				.IDENT	\V04-000\	
				.PSECT	\$CODE\$,NOWRT,2	
FFFF	FFFF	FFFF	00000	.WORD	-1, -1, -1	:
2A	2E	2A	00006	.ASCII	*.*\	:
				.EXTRN	PMS_START_SUB, PMS_END_SUB	
				.EXTRN	COPY_NAME, DIR_ACCESS	
				.EXTRN	PARSE_NAME, DIR_SCAN	
				.EXTRN	NEXT_REC, RETURN_DIR	
				.EXTRN	MARK_DIRTY, NEXT_DIR_REC	
				.EXTRN	REMOVE	
				.ENTRY	FIND, Save R2,R3,R4,R5,R6,R7,R8,R9	: 1082
59	0000G	CF	9E	MOVAB	DIR_SCAN, R9	:
5E	FEE0	CE	9E	MOVAB	-288(SP), SP	:
57	00D8	CA	9E	MOVAB	216(BASE), R7	: 1144
56	00DC	CA	9E	MOVAB	220(BASE), R6	:
55	0C	A6	9E	MOVAB	12(R6), R5	: 1146
		06	DD	PUSHL	#6	: 1169
00000000G	00	01	FB	CALLS	#1, PMS_START_SUB	:
		6A	95	TSTB	(BASE)	: 1175
		28	18	BGEQ	1\$	:
50	08	AC	D0	MOVL	FIB, R0	: 1178
04	A0	02	AE	MNEGW	#2, 4(R0)	:
50	08	AC	D0	MOVL	FIB, R0	: 1179
06	A0	02	AE	MNEGW	#2, 6(R0)	:
50	08	AC	D0	MOVL	FIB, R0	: 1180
	08	A0	B4	CLRW	8(R0)	:
50	08	AC	D0	MOVL	FIB, R0	: 1181
09	A0	01	8E	MNEGB	#1, 9(R0)	:
	04	AC	DD	PUSHL	ABD	: 1182
0000G	CF	01	FB	CALLS	#1, COPY_NAME	:
		04	0004E	RET		: 1177
50	08	AC	D0	MOVL	FIB, R0	: 1190
7E	14	A0	3C	MOVZWL	20(R0), -(SP)	:
51	04	AC	D0	MOVL	ABD, R1	:
50	10	A1	3C	MOVZWL	16(R1), R0	:
	11	A140	9F	PUSHAB	17(R1)[R0]	:
7E	12	A1	3C	MOVZWL	18(R1), -(SP)	: 1189
	90	AD	9F	PUSHAB	NAME_BUFFER	:
	F0	AD	9F	PUSHAB	NAME_DESC	:
0000G	CF	05	FB	CALLS	#5, PARSE_NAME	:
	0C	AC	D5	TSTL	FIND_MODE	: 1198

				04	13	00075	BEQL	2\$		
				01	DD	00077	PUSHL	#1		
				0D	11	00079	BRB	5\$		
		04	F1	AD	E9	0007B	2\$: BLBC	NAME_DESC+1, 3\$	1200	
				50	D4	0007F	CLRL	R0		
				03	11	00081	BRB	4\$		
		50		06	D0	00083	3\$: MOVL	#6, R0		
				50	DD	00086	4\$: PUSHL	R0		
		0000G	CF	08	AC	DD	5\$: PUSHL	FIB	1197	
				02	FB	0008B	CALLS	#2, DIR_ACCESS		
				0C	AC	D5	TSTL	FIND_MODE	1209	
				13	12	00093	BNEQ	6\$		
		50		08	AC	D0	MOVL	FIB, R0	1210	
		OA	15	A0	03	E1	BBC	#3, 21(R0), 6\$		
				50	AC	D0	MOVL	ABD, R0	1211	
				04	A0	B5	TSTW	34(R0)		
				22	01	12	BNEQ	6\$		
					04	000A7	RET			
					58	D4	6\$: CLRL	START_BLOCK	1214	
					53	7C	CLRL	START_VER	1216	
					67	D4	CLRL	(R7)	1217	
				14	A6	D4	CLRL	20(R6)	1218	
				1C	A6	94	CLRB	28(R6)	1219	
		50		08	AC	D0	MOVL	FIB, R0	1225	
A0	AA		08	A0	00	ED	CMPZV	#0, #8, 8(R0), -96(BASE)		
				08	03	12	BNEQ	7\$		
					A0	94	CLRB	8(R0)		
		50		08	AC	D0	7\$: MOVL	FIB, R0	1232	
		50			10	C0	ADDL2	#16, R0		
					60	D5	TSTL	(R0)		
					03	12	BNEQ	8\$		
					00F0	31	BRW	18\$		
					06	EF	8\$: EXTZV	#6, #10, (R0), START_BLOCK	1235	
					00	EF	EXTZV	#0, #6, (R0), START_REC	1236	
					54	D7	DECL	START_REC		
		50		04	AC	D0	MOVL	ABD, R0	1237	
		02		1A	A0	B1	CMPW	26(R0), #2	1238	
					0D	1F	BLSSU	9\$		
		51		18	A0	3C	MOVZWL	24(R0), R1	1240	
				19	A140	9F	PUSHAB	25(R1)(R0)	1239	
		51			9E	3C	MOVZWL	@(SP)+, R1		
					02	11	BRB	10\$		
					51	D4	9\$: CLRL	R1	1238	
		52		22	A0	3C	10\$: MOVZWL	34(R0), R2		
		51			52	D1	CPL	R2, R1		
					03	1B	BLEQU	11\$		
		52			51	D0	MOVL	R1, R2		
					7E	D4	11\$: CLRL	-(SP)	1243	
		51		20	A0	3C	MOVZWL	32(R0), R1	1244	
				21	A140	9F	PUSHAB	33(R1)(R0)		
					52	DD	PUSHL	COUNT	1243	
				6C	AE	9F	PUSHAB	PREV_BUFFER		
				E0	AD	9F	PUSHAB	PREV_DESC		
		0000G	CF		05	FB	CALLS	#5, PARSE_NAME		
					E1	AD	BLBC	PREV_DESC+1, 12\$	1245	
				0818	8F	BF	CHMU	#2072	1246	
					04	00122	RET			

				52	D5	00123	12\$:	TSTL	COUNT	:	1253
				4A	12	00125		BNEQ	13\$	:	
	E1	AD		08	88	00127		BISB2	#8, PREV_DESC+1	:	1256
				54	DD	0012B		PUSHL	START_REC	:	1257
				7E	7C	0012D		CLRQ	-(SP)	:	
				7E	D4	0012F		CLRL	-(SP)	:	
				58	DD	00131		PUSHL	START_BLOCK	:	
			FECO	CF	9F	00133		PUSHAB	P.AAA	:	
			E0	AD	9F	00137		PUSHAB	PREV_DESC	:	
		69		07	FB	0013A		CALLS	#7, DIR_SCAN	:	
6D	FO	AD		03	E0	0013D		BBS	#3, NAME_DESC, 15\$	:	1258
	EO	AD	0100	8F	B0	00142		MOVW	#256, PREV_DESC	:	1261
	E4	AD		03	D0	00148		MOVL	#3, PREV_DESC+4	:	1262
	E8	AD	FEAD	CF	9E	0014C		MOVAB	P.AAB, PREV_DESC+8	:	1263
	EC	AD	8000	8F	B0	00152		MOVW	#-32768, PREV_DESC+12	:	1264
		7E		01	CE	00158		MNEGL	#1, -(SP)	:	1265
			14	A6	DD	0015B		PUSHL	20(R6)	:	
				65	DD	0015E		PUSHL	(R5)	:	
			08	A6	DD	00160		PUSHL	8(R6)	:	
7E		66		01	C3	00163		SUBL3	#1, (R6), -(SP)	:	
				7E	D4	00167		CLRL	-(SP)	:	
			E0	AD	9F	00169		PUSHAB	PREV_DESC	:	
		69		07	FB	0016C		CALLS	#7, DIR_SCAN	:	
				3E	11	0016F		BRB	15\$	:	1267
06	FO	AD		03	E0	00171	13\$:	BBS	#3, NAME_DESC, 14\$	:	1281
	EC	AD	8000	8F	B0	00176		MOVW	#-32768, PREV_DESC+12	:	1282
		7E		01	CE	0017C	14\$:	MNEGL	#1, -(SP)	:	1283
				7E	7C	0017F		CLRQ	-(SP)	:	
				7E	D4	00181		CLRL	-(SP)	:	
				58	DD	00183		PUSHL	START_BLOCK	:	
				7E	D4	00185		CLRL	-(SP)	:	
			E0	AD	9F	00187		PUSHAB	PREV_DESC	:	
		69		07	FB	0018A		CALLS	#7, DIR_SCAN	:	
		1F		50	E8	0018D		BLBS	R0, 15\$	:	
50		66		01	C3	00190		SUBL3	#1, (R6), R0	:	1291
		58		50	D1	00194		CMPL	R0, START_BLOCK	:	
				1E	1A	00197		BGTRU	16\$	:	
				67	D5	00199		TSTL	(R7)	:	1292
				1A	12	0019B		BNEQ	16\$	:	
		7E		01	CE	0019D		MNEGL	#1, -(SP)	:	1293
				7E	7C	001A0		CLRQ	-(SP)	:	
				7E	7C	001A2		CLRQ	-(SP)	:	
				7E	D4	001A4		CLRL	-(SP)	:	
			E0	AD	9F	001A6		PUSHAB	PREV_DESC	:	
		69		07	FB	001A9		CALLS	#7, DIR_SCAN	:	
		08		50	E9	001AC		BLBC	R0, 16\$	:	
53		65		08	C1	001AF	15\$:	ADDL3	#8, (R5), START_VER	:	1298
				67	D6	001B3		INCL	(R7)	:	1299
				03	11	001B5		BRB	17\$	:	1289
		53		65	D0	001B7	16\$:	MOVL	(R5), START_VER	:	1301
		54	08	A6	D0	001BA	17\$:	MOVL	8(R6), START_REC	:	1304
58		66		01	C3	001BE		SUBL3	#1, (R6), START_BLOCK	:	1305
		7E		01	CE	001C2	18\$:	MNEGL	#1, -(SP)	:	1311
			14	A6	DD	001C5		PUSHL	20(R6)	:	
				53	DD	001C8		PUSHL	START_VER	:	
				54	DD	001CA		PUSHL	START_REC	:	
				58	DD	001CC		PUSHL	START_BLOCK	:	

7E	08	AC		04	C1	001CE	ADDL3	#4, FIB, -(SP)		
			F0	AD	9F	001D3	PUSHAB	NAME_DESC		
		69		07	FB	001D6	CALLS	#7, DIR_SCAN		
		16		50	E8	001D9	BLBS	R0, 20\$		
		50		08	AC	D0	001DC	MOVL	FIB, R0	1313
			10	A0	D5	001E0	TSTL	16(R0)		
				05	12	001E3	BNEQ	19\$		
			0910	8F	BF	001E5	CHMU	#2320		1314
					04	001E9	RET			1316
			10	A0	D4	001EA	CLRL	16(R0)		1317
			0930	8F	BF	001ED	CHMU	#2352		1318
					04	001F1	RET			1316
		51		65	D0	001F2	MOVL	(R5), R1		1325
		50		08	AC	D0	001F5	MOVL	FIB, R0	
04	A0	02		06	28	001F9	MOVC3	#6, 2(R1), 4(R0)		
		A1		08	AC	D0	001FF	MOVL	FIB, R0	1326
		50		08	A0	95	00203	TSTB	8(R0)	
				05	12	00206	BNEQ	21\$		
		08	A0	AA	90	00208	MOVB	-96(BASE), 8(R0)		
		50		08	AC	D0	0020D	MOVL	FIB, R0	
		01		08	A0	91	00211	CMPB	8(R0), #1	
				08	12	00215	BNEQ	22\$		
			A0	AA	D5	00217	TSTL	-96(BASE)		
				03	12	0021A	BNEQ	22\$		
			08	A0	94	0021C	CLRB	8(R0)		
		50		08	AC	D0	0021F	MOVL	FIB, R0	1331
		0D		F1	AD	E8	00223	BLBS	NAME_DESC+1, 23\$	1327
08	F0	AD		03	E0	00227	BBS	#3, NAME_DESC, 23\$		1328
		51		08	AC	D0	0022C	MOVL	FIB, R1	1329
		0E		15	A1	E9	00230	BLBC	21(R1), 24\$	
51		66		06	78	00234	ASHL	#6, (R6), R1		1331
		51		67	C0	00238	ADDL2	(R7), R1		
		A0		03	11	00240	MOVAB	-63(R1), 16(R0)		
				10	A0	D4	00242	BRB	25\$	
				04	AC	DD	00245	CLRL	16(R0)	1333
				0C	AE	9F	00248	PUSHL	ABD	1335
				08	AE	9F	0024B	PUSHAB	RESULT_STRING	
					03	FB	0024E	PUSHAB	RESULT_LENGTH	
		0000G	CF	0A	1F	00256	CALLS	#3, RETURN_DIR		1336
		05		6C	91	00253	CMPB	(AP), #5		
				0A	1F	00256	BLSSU	26\$		
		10	BC	6E	D0	00258	MOVL	RESULT_LENGTH, @RESLEN_ARG		1339
14	BC	08	AE	6E	28	0025C	MOVC3	RESULT_LENGTH, RESULT_STRING, @RESULT_ARG		1340
		02		0C	AC	D1	00262	CMPB	FIND_MODE, #2	1348
					2C	12	00266	BNEQ	28\$	
				1A	A6	B5	00268	TSTW	26(R6)	1349
				27	12	0026B	BNEQ	28\$		
		52		08	A6	D0	0026D	MOVL	8(R6), P	1352
		AE		66	D0	00271	MOVL	(R6), VBN		1353
		50		08	AC	D0	00275	MOVL	FIB, R0	1356
		A2		2C	A0	B0	00279	MOVW	44(R0), 2(P)	
					52	DD	0027E	PUSHL	P	1357
		0000G	CF	01	FB	00280	CALLS	#1, MARK_DIRTY		
				04	AE	9F	00285	PUSHAB	VBN	1358
					52	DD	00288	PUSHL	P	
		0000G	CF	02	FB	0028A	CALLS	#2, NEXT_DIR_REC		
		52		50	D0	0028F	MOVL	R0, P		

FIND
V04-000

J 11
16-Sep-1984 00:31:12
14-Sep-1984 12:30:27

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]FIND.B32;1

Page 12
(2)

		50	08	E1	12	00292		BNEQ	27\$		1360
	2C	A0	18	AC	D0	00294	28\$:	MOVL	FIB, R0		1362
		07	0C	A6	B0	00298		MOVW	24(R6), 44(R0)		
				AC	E9	0029D		BLBC	FIND MODE, 29\$		1367
				7E	D4	002A1		CLRL	-(SP)		1368
	0000G	CF		01	FB	002A3		CALLS	#1, REMOVE		
00000000G		00		00	FB	002A8	29\$:	CALLS	#0, PMS_END_SUB		1373
				04	002AF			RET			1375

; Routine Size: 688 bytes, Routine Base: \$CODE\$ + 0009

: 387 1376 1
: 388 1377 1 END
: 389 1378 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	697	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	38	0	1000	00:02.0

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:FIND/OBJ=OBJ\$:FIND MSRC\$:FIND/UPDATE=(ENH\$:FIND)

: Size: 688 code + 9 data bytes
: Run Time: 00:26.2
: Elapsed Time: 00:59.7
: Lines/CPU Min: 3153
: Lexemes/CPU-Min: 39048
: Memory Used: 347 pages
: Compilation Complete

0170

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY